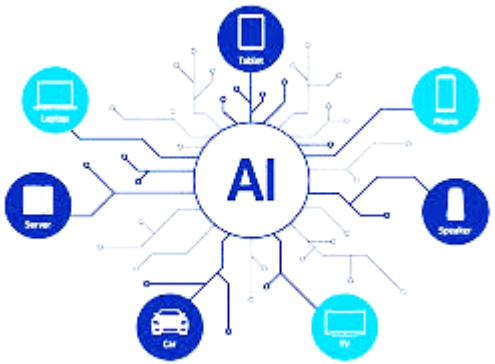
توقع أسعار العملة التركية مقابل الدولار

الإنحدار الخطي

تعلم شجرة القرار

كي ان ان



الهدف

إنشاء نموذج تنبؤ دقيق لسعر العملة التركية مقابل الدولار بناءً على البيانات التاريخية المتاحة.
سيكون لدينا اعتمادًا على تحليل الانحدار الخطي وشجرة القرار لتحليل البيانات وتنبؤ الأسعار

الأدوات:

- مكتبة Pandas NumPy:

سنستخدم مكتبة Pandas لتحليل وتنظيم البيانات وNumPy للعمليات الرياضية والمتقدمة.

- مكتبة Sklearn : سنستخدم Sklearn لتطبيق تقنيات الانحدار الخطي وشجرة القرار. هذه المكتبة توفر واجهة برمجة التطبيقات لتطبيق مجموعة واسعة من خوارزميات تعلم الآلة.
- مكتبة Matplotlib Seaborn:

سنستخدم هذه المكتبات لرسم البيانات والرسوم البيانية التوضيحية لفهم أفضل للتوقعات والنتائج.

- **خطوات المشروع المتوقعة:**

١- جمع البيانات:

سنقوم بجمع البيانات التاريخية لأسعار العملة التركية مقابل الدولار. يمكن الحصول على هذه البيانات من مصادر مختلفة مثل مواقع تبادل العملات الرقمية.

٢- تحليل البيانات:

سنستخدم Pandas لتنظيف وتنظيم البيانات، ثم سنقوم بتحليلها لاستكشاف العلاقات بين متغيرات البيانات المختلفة.

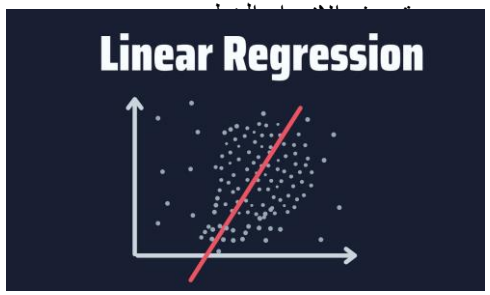
٣- تدريب النماذج:

سنقوم بتقسيم البيانات إلى مجموعات تدريب واختبار ونقوم بتدريب النماذج باستخدام خوارزميات الانحدار الخطي وشجرة القرار.

٤- تقييم النماذج:

سنقوم بتقييم أداء النماذج باستخدام مقاييس مثل معدل الخطأ المتوقع.


DATA
MINING



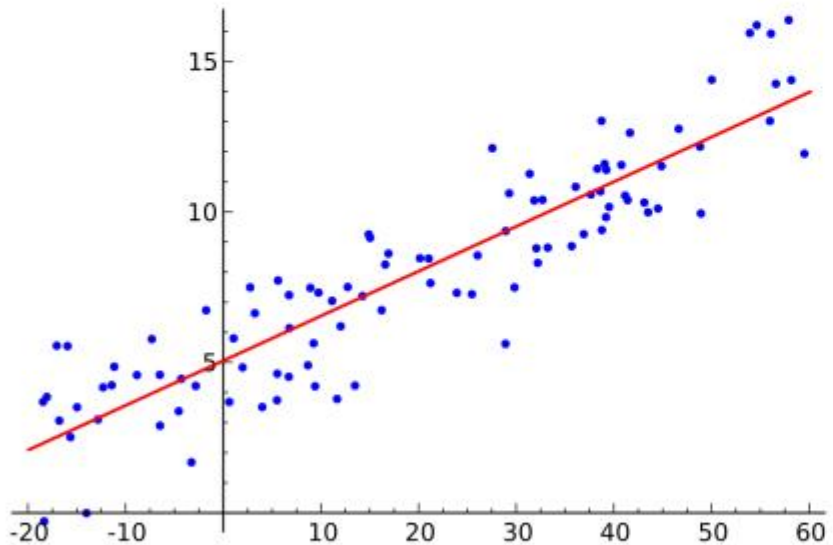
تعريف الإنحدار الخطي

• الانحدار الخطي هو خوارزمية إحصائية تستخدم لنمذجة العالقة بين متغيرين.
تفترض هذه الخوارزمية أن هناك عالقة خطية بين المتغير التابع (المتغير الذي يتم توقعه أو شرحه) ومتغير واحد أو أكثر من المتغيرات المستقلة (تلك المستخدمة إلجراء التنبؤ). الهدف من الانحدار الخطي هو العثور على الخط الانسب الذي يصف العالقة بين المتغيرات. حيث يتم تحديد أفضل خط مناسب بتقليل مجموع الفروق التربيعية بين القيم الفعلية والقيم المتوقعة.

• الانحدار الخطي نوعان: الانحدار الخطي البسيط والانحدار الخطي المتعدد. يتميز الانحدار الخطي البسيط بمتغير مستقل واحد. بينما يتميز الانحدار الخطي المتعدد بالعديد من المتغيرات المستقلة (أكثر من واحد) أثناء البحث عن أفضل خط مناسب.

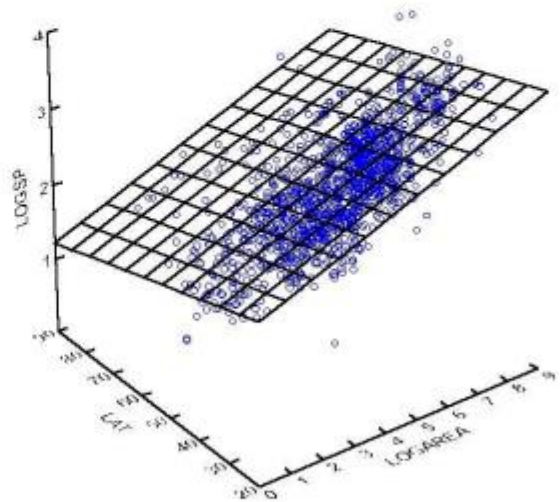
أنواع الإنحدار الخطي

الإنحدار الخطي البسيط



$y = bX + c$

الإنحدار الخطي المتعدد



$y = b_1x_1 + b_2x_2 + \ldots + b_nx_n + c$

شجرة القرار

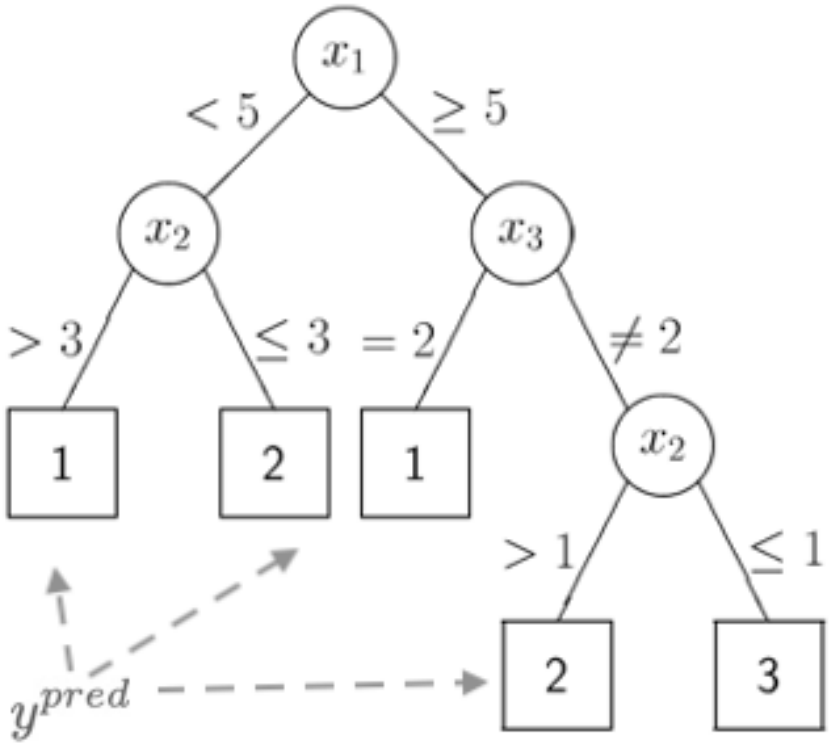
- شجرة القرار (Decision Tree):

شجرة القرار هي نموذج تعلم آلي يُستخدم لأغراض التوقع والتصنيف. يقوم هذا النموذج ببناء شجرة شبيهة بالرسم البياني لاتخاذ القرارات بناءً على ميزات البيانات المدخلة. تُستخدم شجرة القرار في مجالات متعددة، مثل توقع الأسعار، تصنيف الصور، وتحليل النصوص

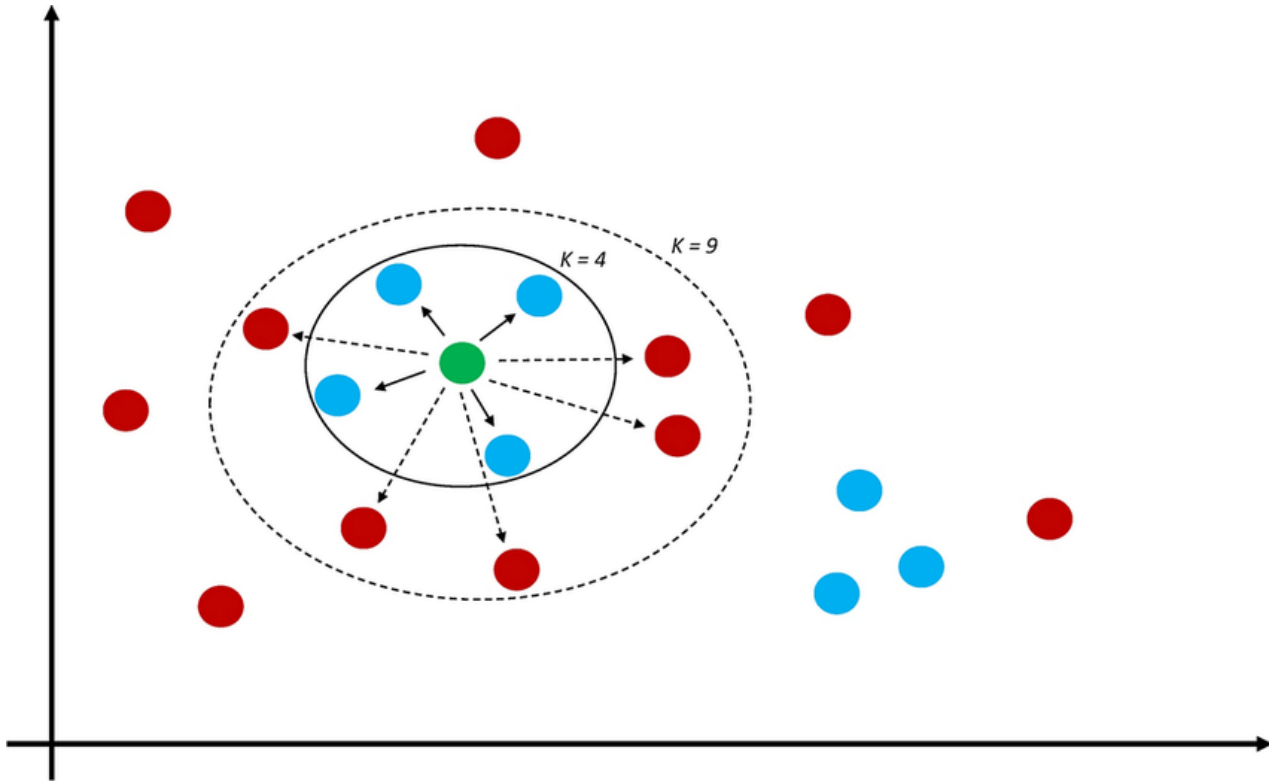
بعد بناء الشجرة، يتم استخدامها للتوقع على مجموعة جديدة منالبيانات.

تبدأ العملية في الجذر، حيث يتم اتخاذ القرار بناءً على السمة الموجودة في الجذر.

يتم اتخاذ القرارات المتتالية في كل عقدة داخلية حتى الوصول إلى عقدة نهائية (ورقة). عند الوصول إلى عقدة نهائية، يتم إصدار التنبؤ أو القرار النهائي.



تعلم الجار الأقرب (K-Nearest Neighbors)

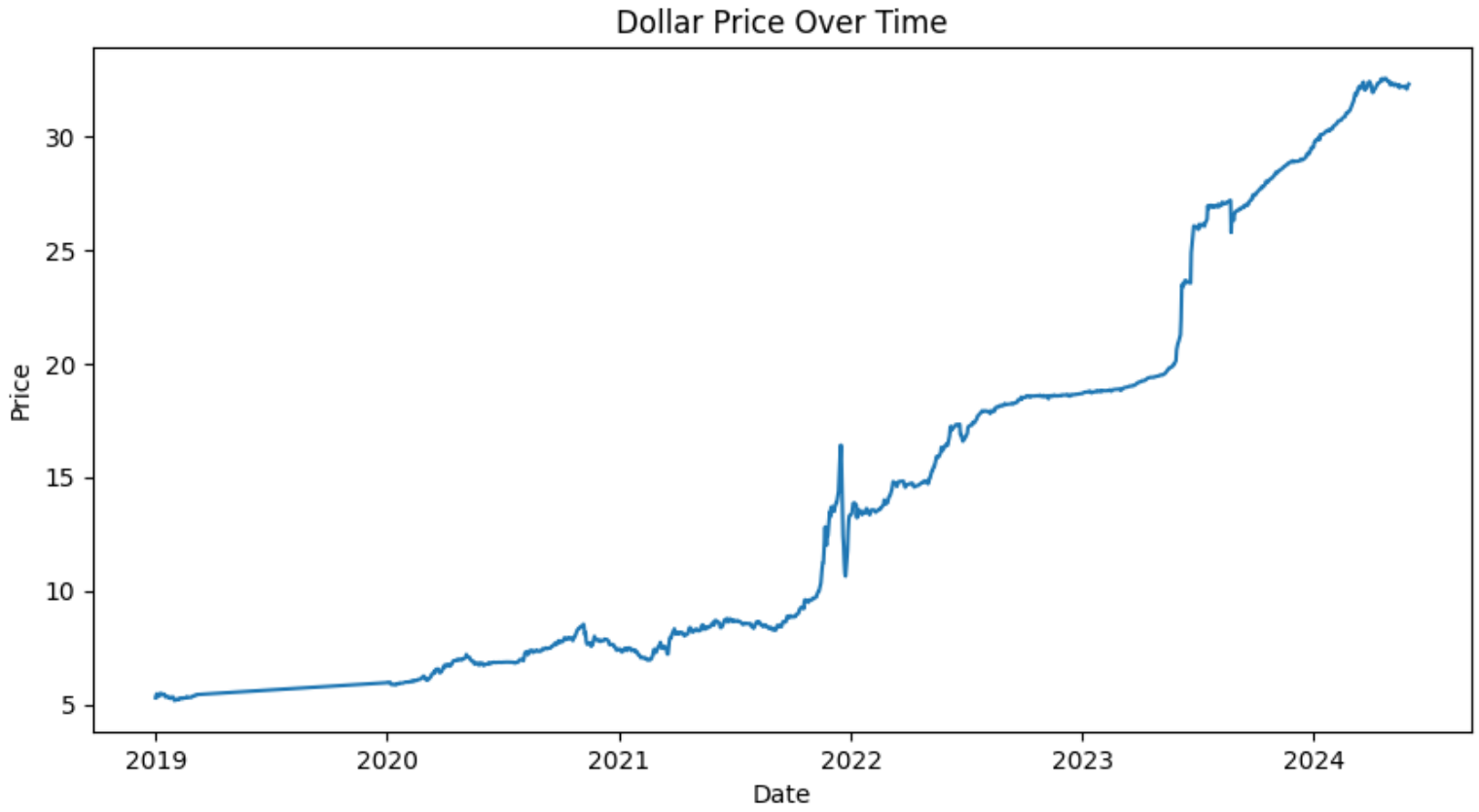


- تعلم الجار الأقرب (K-Nearest Neighbors) هو نموذج تعلم آلي يُستخدم للتنبؤ بقيمة جديدة بناءً على قيم مماثلة للبيانات المتاحة. يعتمد هذا النموذج على مفهوم القرب، عندما يتلقى النموذج نقطة جديدة، يقوم بالبحث عن النقاط الأقرب إليها في مجموعة التدريب ويستخدم قيمها لتقديم التوقع.

- الخطوة الأولى هي الحصول على البيانات التي سيتم تحليلها. في هذه الحالة، لدينا ملف CSVيحتوي على تاريخ وسعر الليرة التركية مقابل الدولار. مصدر الملف

Date	Price	Open	High	Low	Vol	Change %
05/30/2024	32.3217	32.2648	32.3323	32.1917		0.20%
05/29/2024	32.258	32.2470	32.3050	32.1575		0.03%
.....	.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....	.....
01/01/2019	5.2933	5.2896	5.2933	5.2540		0.09%

مخطط سعر العملة التركية مقابل الدولار منذ عام ٢٠١٩



برنامج تحليل وتنبؤ بسعر الليرة التركية مقابل الدولار الأمريكي باستخدام مكتبات بايثون المختلفة مثل:

Pandas لمعالجة البيانات.

Numpy للتعامل مع المصفوفات والأعداد.

Matplotlib لرسم المخططات البيانية.

mean_squared_error لحساب خطأ التربيع المتوسط لتقييم النموذج.

Sklearn لإستخدام وتدريب النماذج

إنشاء كلاس UsfToTRY للتنبؤ بالسعر

- قراءة البيانات:

```
class UsdToTRY:
    def __init__(self) :
        # self.data = pd.read_csv('USD_TRY Historical Data.csv')
        self.data = pd.read_csv('USD_TRY Historical 6-years.csv')
```

- فحص البيانات: لعرض الصفوف الأولى من البيانات، الإحصائيات الأساسية، والتحقق من القيم المفقودة.

```
def checkData(self):
        # Display the first few rows

        print(f'head {"-"*100}\n{self.data.head()}')

        # Basic statistics
        print(f'describe{"-"*100}\n{self.data.describe()}')

        # Check for missing values
        print(f'is Null {"-"*100}\n{self.data.isnull().sum()}')
```

معالجة البيانات:

تحويل عمود Dateإلى صيغة تاريخية.

إزالة القيم المكررة.

ترتيب البيانات حسب التاريخ.

استخراج اليوم، الشهر، والسنة من عمود التاريخ.

إزالة عمود .Volالفارغ.

إزالة الصفوف التي تحتوي على قيم مفقودة.

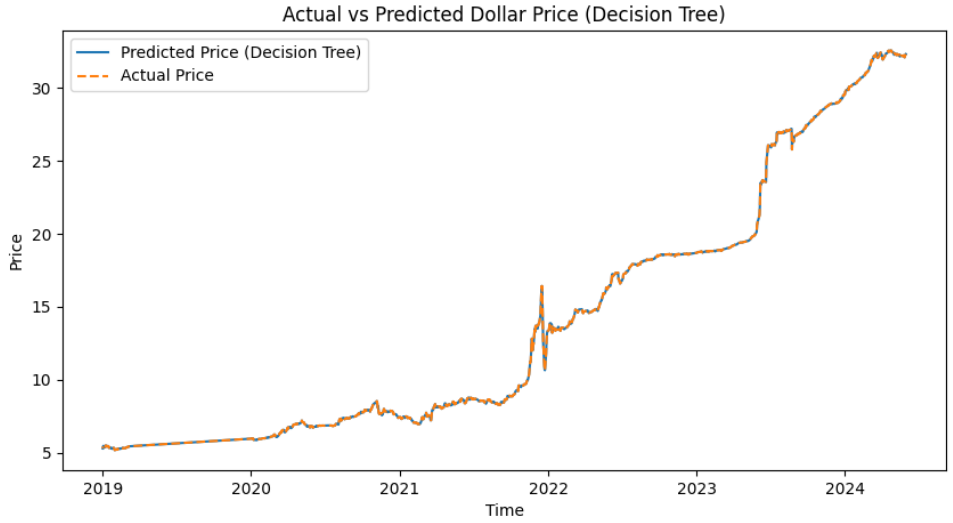
تعيين المتغيرات المستقلة (X) والمتغير التابع (Y)

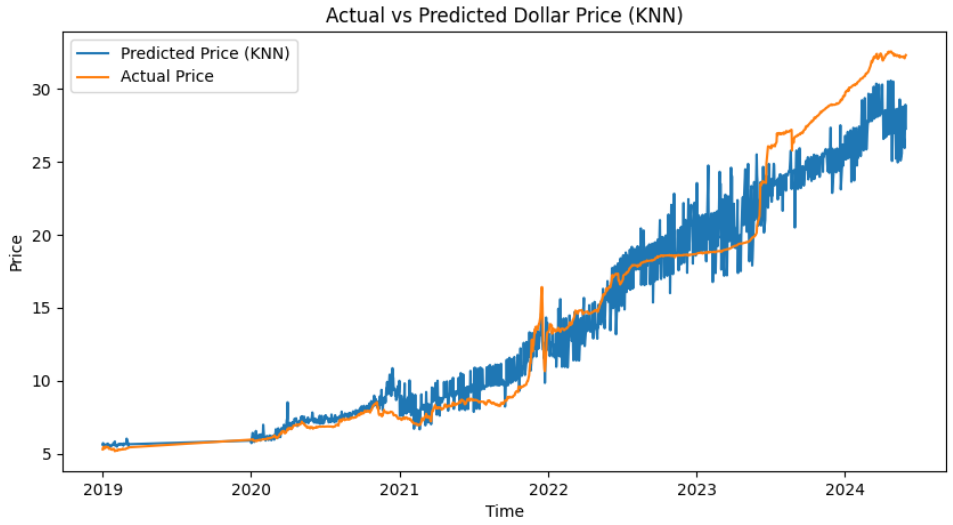
```
def dataPreprocessing(self):
        self.data['Date'] = pd.to_datetime(self.data['Date'])



        self.data = self.data.drop_duplicates(subset=['Date'])
        self.data =
self.data.sort_values(by='Date',ascending=False)
        self.data['day']=self.data['Date'].dt.day
        self.data['month']=self.data['Date'].dt.month
        self.data['year']=self.data['Date'].dt.year
        #drop Empty Column
        self.data=self.data.drop('Vol.', axis=1)
        self.data.dropna(inplace=True)
        self.X=self.data[['day','month','year']]
        self.Y=self.data['Price']
```

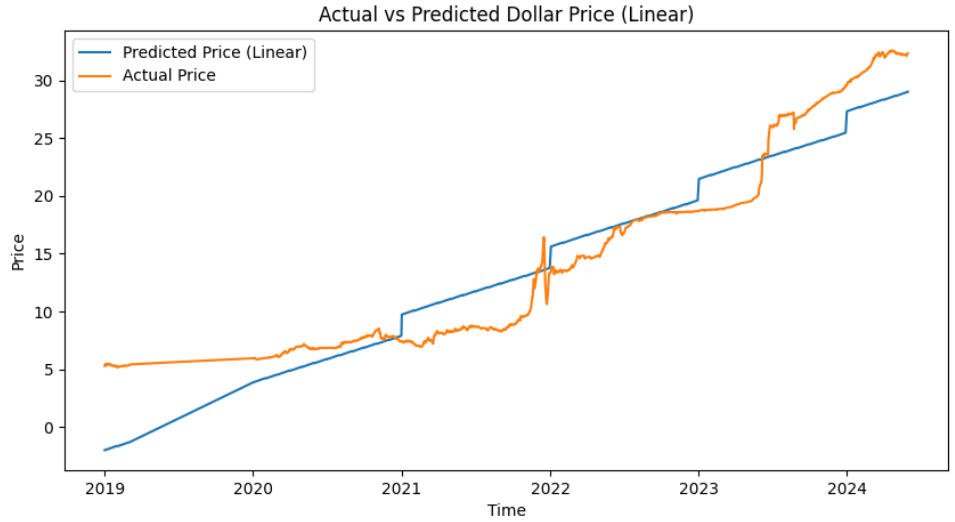
بناء وتدريب النماذج

```
  def buildAndTrainTheLinearModel(self):
        self.linear_model = LinearRegression()
        self.linear_model.fit(self.X, self.Y)

  def buildAndTrainTheDecisionTreeModel(self):
        self.decision_tree_model = DecisionTreeRegressor()
        self.decision_tree_model.fit(self.X, self.Y)

    def buildAndTrainTheKNNModel(self):
        self.knn_model = KNeighborsRegressor(n_neighbors=7)
        self.knn_model.fit(self.X, self.Y)
```

```
def evaluateTheModel(self):
        # Make predictions
        # print(self.X_test)
        y_pred = self.model.predict(self.data[['day','month','year']])
        # Calculate the mean squared error
        mse = mean_squared_error(self.Y, y_pred)
        print(f'Mean Squared Error: {mse}')
        # Plot actual vs predicted prices
        plt.figure(figsize=(10,5))
        plt.plot(y_pred, label='Predicted Price')
        plt.plot(self.Y, label='Actual Price')
        plt.title('Actual vs Predicted Dollar Price')
        plt.xlabel('Time')
        plt.ylabel('Price')
        plt.legend()
        plt.show()
```


Actual vs Predicted Dollar Price (Decision Tree)
Predicted Price (Decision Tree)
Actual Price
Price
Time
5
10
15
20
25
30
2019
2020
2021
2022
2023
2024


Actual vs Predicted Dollar Price (KNN)
Predicted Price (KNN)
Actual Price
Price
Time
5
10
15
20
25
30
2019
2020
2021
2022
2023
2024


Actual vs Predicted Dollar Price (Linear)
Predicted Price (Linear)
Actual Price
Price
Time
0
5
10
15
20
25
30
2019
2020
2021
2022
2023
2024